1. Module number	*SET09117*
2. Module title	*Algorithms and Data Structures*
3. Module leader	*Neil Urquhart*
4. Tutor with responsibility for this Assessment Student's first point of contact	*Neil Urquhart*
5. Assessment	*Vehicle Routing: Code and report.* *A demonstration of your solver must also be made*
6. Weighting	*40% of module assessment.*
7. Size and/or time limits for assessment	*Submitted code should be succinct and well commented. The indentation must be correct.*
8. Deadline of submission Your attention is drawn to the penalties for late submission	*Submission by the end of week 11.* *Demonstrations should take place in class during week 11.*
9. Arrangements for submission	*Hand in via moodle*
10. Assessment Regulations All assessments are subject to the University Regulations**.**	
11. The requirements for the assessment	*Complete the tasks given. Submit a report of your implementation. The report should include your source code.*
12. Special instructions	*The report must be submitted using the moodle page*
13. Return of work	*Your feedback and grade will be returned via moodle.*
14. Assessment criteria	*You will be assessed on* • *your use of appropriate algorithms and data structures* • *your ability to conduct an experiment and present the results*

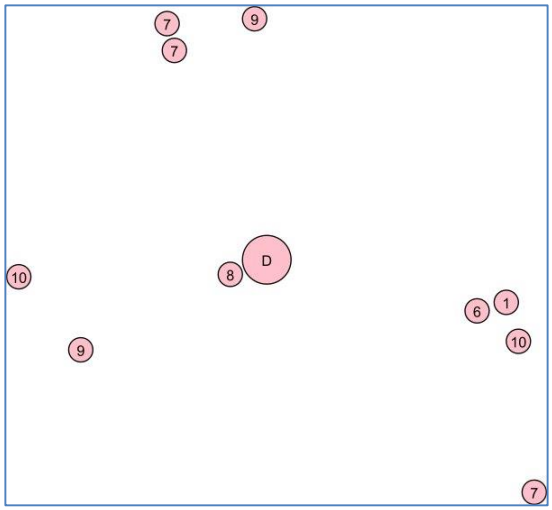
SET09117 Algorithms and Data Structures Coursework 2015

Introduction

A company wish to deliver goods from their warehouse to their customers. You are required to implement a vehicle routing algorithm for them. They have a set of sample problems for you to work with and they have of answers to the sample problems.
You must implement a vehicle routing algorithm (e.g. Clarke-Wright) and produce solutions to the sample problems. You will prepare a report which outlines your approach and how your solutions compare to the example solutions provided. You must demonstrate that your solution is correct.

The problem

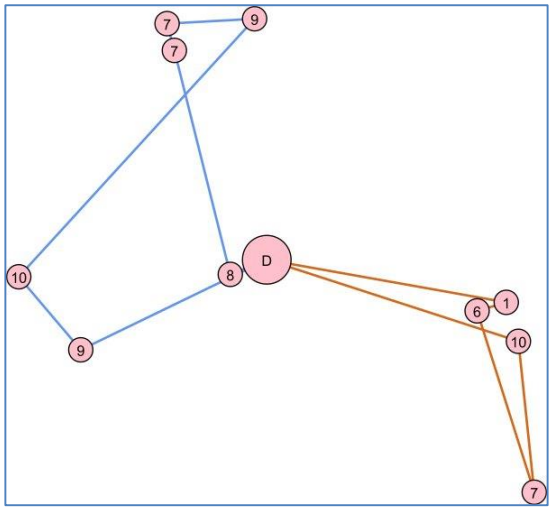
Each problem has a depot and a set of customers, they have a location defined by XY coordinates, a typical problem might look like this:



The above problem has 10 customers who require deliveries from the depot 'D' to be made by truck.

Each customer has a requirement, this is a number that represents the quantity of goods they require. Each truck has a capacity (all trucks in this problem have the same capacity). To solve the problem customers must be grouped together so that each group can be served by a single truck which begins and ends its journey at the depot. The total of all the customer demands within a group must not exceed the capacity of the truck. Each solution has a cost based on the number of trucks used and the distance travelled.

A possible solution to the above problem is shown below, in this solution 2 trucks are used, one serving 6 customers and one serving 4.



Provided Files

Problem instances and solutions –
contained in SET09117data.zip:

*prob.csv	problem instances First line gives cords of depot & max capacity of a truck Subsequent triples are coords of and the requirement
*cwsn.csv	Solution as generated by our Clarke-Wright algorithm Each line has a list of triples indicating a route
*dmsn.csv	Poor but fast solution to the problem. These files are provided to give examples of solutions for you to benchmark your work against.
*.svg	A visualisation of a problem or solution

Java Source code

Customer.java	A class representing a customer (also misused for the depot)
VRProblem.java	Represents a single vehicle routing problem
VRSolution.java	Represents a single solution (or partial solution) to the VRP. Also contains a very simple solver – see below

Verifier

Verifier.jar	A utility which will check the correctness of a solution, confirm the cost and produce .SVG graphic,

Hints to get you started.

1. Unzip SET09117 CW 2014 Test data.zip, this will provide a large sample of problem instances and solutions as outlined above

2. The following example Java code shows how to load problem instance and create a solution using the existing solver. The existing solver is rubbish – it allocates each customer to an individual route!

```
public static void main(String[] args) throws Exception  {


        VRProblem p = new VRProblem("rand00040prob.csv");
        //Load problem

        VRSolution s = new VRSolution(p);
        //Create blank solution

        s.oneRoutePerCustomerSolution();
        //Use the existing problem solver to build a solution

        System.out.println("Cost =" + s.solnCost());
        //Print out the cost of the solution


        s.writeOut("MySolution.csv");
        //Save the solution file for verification

        s.writeSVG("rand00040prob.svg", "MyPictureSolution.svg");
        //Create pictures of the problem and the solution
}
```

Try implementing the above and creating solutions for the supplied files. This code snippet below shows how you can measure the speed of the solver:

```
        double startTime = System.currentTimeMillis();
        //Time started
        s.oneRoutePerCustomerSolution();
        //Use the existing problem solver to build a solution

        double endTime = System.currentTimeMillis();
        //Time finished

        System.out.println("The time taken was " + (endTime - startTime));
```

Make sure that you can use the supplied code and the examples above to create solutions to the supplied problems.

3. When you create solutions to the problems it is important to be able to verify that they are valid (i.e. no missed out customers etc!). We provide a simple JAR file that contains a verifier that will check your solutions for you. You can run the verifier from a command line as follows:

```
C:\A_DS\CW>java -jar Verify.jar rand00010prob.csv rand00010cwsn.csv
```

```
Solution is valid
Cost:1576.5325944728006
```

The first input to Verify.Jar is the problem instance, the second is the solution. If the solution is valid a cost will be printed, if there is an error in the solution an error message will be printed. The correct usage is

Java –jar Verify.jar <problem.csv> <solution.csv>

e.g.

```
C:\A_DS\CW>java -jar Verify.jar fail00002prob.csv fail00002soln.csv
********FAIL Route starting Point2D.Double[200.0, 200.0] is over capacity 70
```

Verifier will also output 2 .SVG files giving a graphical representation of the problem and solution. They are named as per the problem and solution files, but with .SVG appended.

4. Make sure that you understand how the OneRoutePerCustomerSolution() method works.

5. To get started, implement a new solver within the VRSolution class, e.g. (VRSolution.mySolver()) When you write your solution files, rename them along the lines of *mySol.CSV to differentiate them from the supplied solutions.

Use of languages other than Java

You are welcome to work in any of the following languages rather than Java

- C/C++
- Python
- C#

You will need to write your own equivalents of VRProblem.java and VRSolution.java . Your solver must be able to read and write the .CSV files so that your problems can be verified and costed using Verifier.jar (no matter what language is used, you must use Verifier.jar to check your solutions.
The format of the problem files is as follows:

<depotX>,<depotY>,<truck capacity>
<customerX>,<customerY>,<requirement>
<customerX>,<customerY>,<requirement>
<customerX>,<customerY>,<requirement>
…
The first line defines the depot position and the capacity of the trucks that operate from it. Subsequent lines identify the location of each customer and the quantity of goods required by them.

The format of the solution files is also a .CSV file using one line per truck, with all of the customers within that route on that line. As in the input file, wach customer has 3 values (x,y,capacity):

<customer1X>,<customer1Y>,<requiremen1t>,<customer2X>,<customer2Y>,<requirement2>,<customer3X>,<customer3Y>,<requiremen3t>
<customer4X>,<customer4Y>,<requiremen4t>,<customer5X>,<customer5Y>,<requirement5>

The above example shows a solution involving 5 customers across 2 routes.

Task

You must construct a solver for the VRP problem, if using Java you should add it to VRSolution.java. If you are using another solver you will need to create the solver from scratch. You may wish to use the ClarkeWright algorithm. You should solve the problem instances provided in the .ZIP archive and compare your results to those obtained using the oneRoutePerCustomerSolution and the CWSN.CSV solutions supplied as benchmarks.

You should detail the above in a report to be submitted to Moodle as per below.

Report

Your report should have the following headings:

- **Introduction**
  - Problem summary including any limitations of your solution.
- **Method**
  - Description of the experiment that you conducted.
- **Results**
  - Tables and charts that show the performance of your solution (how long does it take to run your algorithm).
  - Demonstration that your solution is valid.
  - Demonstration of the quality of your solution (what is the cost).
- **Conclusions**
  - Summary of your results.
  - Reflection on your performance on this assessment.
- **Appendix**
  - Source code of your solution

Notes on the report:

- The reports must be written in third person.
- The result should presented as a table of figures *and* as a chart. Only averaged results should be shown. Charts must have a title and axis labels for x and y.
- A reasonable number of decimal places should be presented.

SET09117 Coursework assessment sheet.

Element	Comments	Mark
Introduction	Report includes: • A description of the algorithm used with references • An analysis of the expected performance of the algorithm	/20
Experimental method	Report includes: • The methodology used • Steps taken to ensure the accuracy and repeatability of the results	/10
Experimental result	Report includes: • Tables showing the timings for a variety of data sets • Charts that illustrate the run time against data set size	/20
Conclusions and reflections	Report includes: • An explanation for the results obtained • Reflections on how reliable the results are • Reflections on the value of the new implementation	/20
Source code (Appendix)	Implementation includes: • Use of appropriate data structures • Useful comments	/30

NAME: _____

MATRIC: _____

TOTAL: _____

Please note that a successful demonstration of your software must be undertaken in order for this mark to be submitted. Failure to demonstrate or to demonstrate an unsatisfactory solution will result in your mark being reduced.